Preprints of WODES 2018
May 30 - June 1, 2018
Sorrento Coast, Italy

Opacity with powerful attackers

Loïc Hélouët * **Hervé Marchand** * **Laurie Ricker** **

* *INRIA Rennes (e-mail: surname.name@inria.fr).*
** *Mount Allison University, (e-mail: lricker@mta.ca)*

Abstract: This paper considers state-based opacity in a setting where attackers of a secret have additional observation capacities allowing them to know which inputs are allowed by a system. We show that regular opacity is decidable in this setting. We then address the question of controlling a system so that it becomes opaque, and solve this question by recasting the problem in a game setting.

1. INTRODUCTION

Language-based security properties, such as non-interference or opacity, are a way to guarantee that a system fulfills some privacy requirements. The original work of Goguen and Meseguer (1982) is motivated by security issues: given a set of confidential actions performed only by users with high credentials, other users should not be able to infer from their observation whether a "high" action has occurred or not. The formal definition of non-interference states that occurrences of high actions should not affect what other users can *see or do*. Implicitly, this supposes that users have *capabilities* that allow them to check whether some actions are feasible, and can also be used to gain information about the current state of the system.

Non-interference is a limited class of properties: it only addresses questions of leakage from a high to a low level and, in the case of language-based interference, on the occurrence of high actions. However, the properties to protect are not always definable in terms of occurrences of actions from a given subset. A larger setting called *opacity* has been proposed (Bryans et al., 2005) to consider hiding more general properties of runs. This notion formalizes the absence of information flow or, more precisely, the impossibility for an attacker to infer the truth of a predicate ϕ (it could be the occurrence in the system of some particular sequences of events, or the fact that the system is in some particular secret configuration). If the truth of this predicate can be inferred by an attacker, based on its knowledge of the system and the information deduced from its observations, then there is an information flow and the secret is revealed.

This paper considers state-based opacity in a setting where attackers partially observe a system and, in addition to their observations, can query the system to discover the accepted inputs at the current state. In some sense, giving this additional power to attackers reintroduces the notion of capabilities from Goguen and Meseguer (1982). We provide several different models of attackers, ranging from the usual attackers that can only observe a system and try to determine whether the system is currently in a sensitive secret state, to very powerful attackers that know at every instant the possible inputs accepted by the system. We then show how state-based opacity with powerful attackers can be checked. Lastly, we consider opacity enforcement: We bring back this question to the computation of a controller that forbids some inputs to the system to prevent an information flow. We show that opacity enforcement can be modeled as a game in which the controller tries to prevent an information flow, while the rest of the system may systematically perform the worst choices, allowing the attacker to gain information. An extended version (including proofs) is available at: `https://hal.inria.fr/hal-01738169`

2. PRELIMINARIES

Let Σ be an alphabet. A *word* $w = a_1 \dots a_n$ is an element of Σ^*. We denote by ϵ the empty word. Given a subset $\Sigma' \subseteq \Sigma$ and a word $w = a_1 \dots a_n \in \Sigma^*$, the projection of w onto Σ' is the word $P_{\Sigma'}(w)$ obtained by erasing from w all letters from $\Sigma \setminus \Sigma'$. Similarly, we denote by $P^{-1}_{\Sigma'}(\mu)$ the *inverse projection* of μ, i.e., $P^{-1}_{\Sigma'}(\mu) = \{w \in \Sigma^* \mid P_{\Sigma'}(w) = \mu\}$.

Definition 1. A *Labelled Transition System* G is a tuple (Q, q_o, Σ, δ) where Q is a set of states with a distinguished element q_o called *the initial state*, Σ is the set of *events* of G, $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation.

The *runs* of G are sequences of transitions of the form $\rho = q_0 \xrightarrow{a_1} q_1 \dots \xrightarrow{a_n} q_n$ such that $(q_i, a_{i+1}, q_{i+1}) \in \delta$ for every $i \in 0 \dots n-1$. The word associated with a run ρ is the sequence of letters $w = a_1 \dots a_n$ (we also say that w labels run ρ). We denote by $Run(G)$ the set of runs of G, and by $\mathcal{L}(G)$ the (regular) language of words that label runs of G. Furthermore, we assume that G is deterministic, i.e., $\forall q \in Q, \forall a \in \Sigma$, $|(q, a, q')| \leq 1$. Note that as G is deterministic, for every word $w \in \mathcal{L}(G)$ there is a unique run ρ labeled by w. Thus, the map $l : Run(G) \to \mathcal{L}(G)$ that associates runs with their labelings is a bijection.

We consider systems interacting with their environment, i.e., performing inputs and outputs. We assume that Σ is partitioned between the set of input events $\Sigma_?$, the set of output events $\Sigma_!$ and the set of internal events Σ_I. For a given state $q \in Q, \Gamma(q) = \{a \in \Sigma \mid \exists q', (q, a, q') \in \delta\}$ denotes the active event set at q. Similarly, $\Gamma_?(q)$ and $\Gamma_!(q)$ denotes the set of input (respectively output) events that are admissible at state q. Given a set of states $X \subseteq Q$ and a set of input events $\Sigma' \subseteq \Sigma_?$, let $\Gamma_?(X, \Sigma') = \{q \in X \mid \Gamma_?(q) = \Sigma'\}$, i.e., the states of X that have exactly Σ' as active input event set.

2.1 General notion of opacity

We consider a setting where a system should not leak any secret information that might be contained in the

actual run of the system. Such a secret can be depicted as a predicate ϕ over runs of the system, as in Bérard et al. (2015). In this paper, we consider that the secret is represented as a subset of states $S \subseteq Q$. States of S can represent states in which a system is vulnerable to attacks, or states in which some confidential information is in use, and is not yet declassified. We denote by $Run_S(G)$ the set of runs of G ending in a state $q \in S$. We consider that attackers are agents of the system, which are equipped with observation and capabilities. They collect information online with executions of the system and try to infer *with certainty* whether the system's current state belongs to S.

Definition 2. (Attackers). An *attacker* of G is a map $\mathcal{A} : Run(G) \to ((\Sigma \cup \{\epsilon\}) \times O)^*$, where O is a finite alphabet of observed facts obtained through the use of capabilities. We further assume that attackers can be defined inductively, and from their final transitions: for a run $\rho = q_0 \dots q_{n-1} \xrightarrow{a_n} q_n$, $\mathcal{A}(\rho) = \mathcal{A}(q_0 \dots q_{n-1}).obs_{\mathcal{A}}(a_n, q_{n-1})$, where $obs_{\mathcal{A}} : \Sigma \times Q \to (\Sigma \times O) \cup \{\epsilon\}$ defines the observation $obs_{\mathcal{A}}(a, q)$ of the attacker when the system fires a from q.

The definition of attackers above is compositional, i.e., if $\rho = \rho_1.\rho_2$ then $\mathcal{A}(\rho) = \mathcal{A}(\rho_1).\mathcal{A}(\rho_2)$, and transition-based, i.e., for a particular transition (q, a, q'), the observation is of the form (a, o) regardless of its position in a run. Note that an attacker is not necessarily aware of every move of the system as $obs_{\mathcal{A}}(a, q) = \epsilon$ for some actions and states, thus $|\mathcal{A}(\rho)| \leq |\rho|$. Let μ be an observation of the attacker, then the set of runs compatible with μ w.r.t. $\mathcal{A}$ is the set $[\mu]_{\mathcal{A}} = \{\rho \in Run(G) \mid \mathcal{A}(\rho) = \mu\}$.

Definition 3. (Opacity). A secret S is *opaque* w.r.t. $\mathcal{A}$ iff $\forall \rho \in Run_S(G)$ there exists a run ρ' in $[\mathcal{A}(\rho)]_{\mathcal{A}}$ such that $\rho' \notin Run_S(G)$.

Intuitively, opacity says that an attacker is not able to infer that a run is currently in a secret state based on its observation. Definition 2 allows us to decouple the information observed from transitions fired during a run, and information acquired from visited states. Though this definition is very generic, it encompasses many types of attackers, starting with the standard ones without capabilities, and whose observations are limited to a subset of actions $\Sigma_o \subseteq \Sigma$. An attacker observes the system, and builds from its observation a *belief*, i.e., an estimation of a set of states in which the system could be.

Upon the arrival of a new observation, this belief can be updated as follows:

Definition 4. (Belief update). Let G be a system, $\mathcal{A}$ be an attacker, $X \subseteq Q$ be a belief, and (a, o) be an observation. We denote by $\delta_{\mathcal{A}}(X, (a, o))$ the set of states X' such that there exists a run $\rho_{x,x'}$ from a state $x \in X$ to a state of $x' \in X'$, and $\mathcal{A}(\rho_{x,x'}) = (a, o)$. Belief updating extends to words and we can define a function $\Delta_{\mathcal{A}}$ as: $\Delta_{\mathcal{A}}(X, \epsilon) = X$, $\Delta_{\mathcal{A}}(X, w.(a, o)) = \delta_{\mathcal{A}}\left(\Delta_{\mathcal{A}}(X, w), (a, o)\right)$.

Definition 5. (Observers). Let $G = (Q, q_o, \Sigma, \delta)$. The observer of an attacker $\mathcal{A}$ is the deterministic automaton $Det_{\mathcal{A}}(G) = (\mathcal{X}, X_0, \Sigma_{\mathcal{A}}, \delta_{\mathcal{A}})$, with $\mathcal{X} = 2^Q$, $X_0 = \{q_0\}$ [1] and $\Sigma_{\mathcal{A}} = \Sigma_o \times O$.

[1] The approaches in this paper work with sets of possible initial states, allowing attackers to start from beliefs that are not singletons.

The construction of $\delta_{\mathcal{A}}$ depends on the power given to the attacker. However, with the restrictions given to attackers, $Det_{\mathcal{A}}(G)$ is always a finite-state machine. We denote by $Tr_{\mathcal{A}}(G) = \mathcal{L}(Det_{\mathcal{A}}(G))$, the language generated by $Det_{\mathcal{A}}(G)$. The classical notion of opacity can be generalized to observers with capabilities as follows:

Proposition 1. Secret S is *opaque* w.r.t. G and $\mathcal{A}$ iff $\forall \rho \in Run_S(G)$, $\Delta_{\mathcal{A}}(X_0, \mathcal{A}(\rho)) \not\subseteq S$.

Proposition 1 says that a system is opaque if an attacker is not able to infer that a run is currently in a secret state by computing the beliefs obtained by synchronizing its observer automaton with the currently observed run. The verification of opacity is then reduced to the question of reachability of $F = 2^S$ in $Det_{\mathcal{A}}(G)$. This verification is PSPACE-complete (Cassez et al., 2012) in the standard setting where attackers observe only a subset of the labels attached to transitions (defined as attacker $\mathcal{A}_1$ later in this section). As G is deterministic, Proposition 1 can be rephrased as a property of $Det_{\mathcal{A}}(G)$. Furthermore, the PSPACE-completeness proof easily extends to the more general setting.

Corollary 1. (of Proposition 1). Secret S is *opaque* w.r.t. G and $\mathcal{A}$ iff $\forall \mu \in Tr_{\mathcal{A}}(G)$, $\Delta_{\mathcal{A}}(X_0, \mu) \not\subseteq S$.

Corollary 2. Opacity of a system w.r.t. an attacker $\mathcal{A}$ is PSPACE-complete.

2.2 Attacker 1: Standard observer

A standard setting in opacity is to assume that the attacker $\mathcal{A}_1$ observes the system through an interface (a subset of its actions). The attacker does not get information from the current state of the system, but is only aware of the last observable event triggered by the system. This can be modeled in our generic setting, by defining a single observed fact $\bullet$ (i.e., $O = \{\bullet\}$) and a function $obs_{\mathcal{A}_1}$:

$$obs_{\mathcal{A}_1}(a, q) = \begin{cases} \epsilon & \text{if } a \notin \Sigma_o, \\ (a, \bullet) & \text{otherwise.} \end{cases}$$

Given a sequence w of G, the observation of the attacker $\mathcal{A}_1$ is given by the natural projection $P_{\Sigma_o}(w)$. We denote by $Tr_{\mathcal{A}_1}(G) = P_{\Sigma_o}(\mathcal{L}(G))$ this set of observations. Given an observation $\mu \in Tr_{\mathcal{A}_1}(G)$, we denote by $[\![\mu]\!]_{\mathcal{A}_1} = P^{-1}_{\Sigma_o}(\mu) \cap \mathcal{L}(G) \cap \Sigma.\Sigma_o^*$ the set of sequences compatible with μ that end with an observable event. Given an observation $\mu \in Tr_{\mathcal{A}_1}(G)$, we denote by $\Delta_o(X_0, \mu) = \{\delta(q_0, [\![\mu]\!]_{\mathcal{A}_1})\}$, the set of states in which the system can be after the observation of the trace μ, with $X_o = \{q_0\}$. Slightly abusing our notation, we use $[\![a]\!]_{\mathcal{A}_1}$ to denote the words for which projection on Σ_o is a and that end with action a. Similarly, for any subset $X \in 2^Q$ and any observable event $a \in \Sigma_o$, $\Delta_o(X, a) = \{q' \in Q \mid \exists q \in X, w \in [\![a]\!]_{\mathcal{A}_1}, q' = \delta(q, w)\}$ denotes the set of states in which the system can be after the observation of a, knowing that the belief of the set of states was $X \in 2^Q$. The secret S is not opaque w.r.t. $\mathcal{A}_1$ whenever there exists $\mu \in Tr_{\mathcal{A}_1}(G)$ such that $\Delta_o(X_o, \mu) \subseteq S$ and the verification is reduced to a reachability problem as described in Cassez et al. (2012).

3. ATTACKERS WITH ADDITIONAL CAPABILITIES

The attacker $\mathcal{A}_1$ is the standard notion of a passive observer that is only informed of a subset of the system's actions. However, this model gives very little discriminating power to attackers. In this section, we propose several

models of attackers that can collect additional information along runs of the system.

3.1 Attacker 2: an input-aware attacker

Let us first define an attacker $\mathcal{A}_2$ that is aware of all the input events it might execute at any time during a run of the system. We assume that the set of inputs that $\mathcal{A}_2$ can observe at any instant is a set $\Sigma_? \subseteq \Sigma_o$, and that this set of available inputs is returned instantaneously after each move as an answer γ, with $\gamma \subseteq \Sigma_?$. We still consider that some actions remain invisible to the attacker. Hence, after the execution of each action a of the system:

- if $a \in \Sigma_o$, the attacker receives the pair (a, γ);
- if $a \in \Sigma_{uo}$, the attacker receives the pair (ϵ, γ).

Note that each time an event is triggered in the system, $\mathcal{A}_2$ is now aware of this occurrence, which can be seen as a "tick" for the attacker whenever the event is not observable. Within this setting, $O_2 = 2^{\Sigma_?}$ and the function $obs_{\mathcal{A}_2}$ is given by

$$obs_{\mathcal{A}_2}(a, q) = \begin{cases} (a, \Gamma_?(\delta(q, a))) & \text{if } a \in \Sigma_o, \\ (\epsilon, \Gamma_?(\delta(q, a))) & \text{otherwise.} \end{cases}$$

The map $\mathcal{A}_2$ directly follows from the definition of $obs_{\mathcal{A}_2}$. Note that even if $\mathcal{A}_2$ observes possible inputs in the current state, some ambiguity remains on the current state, and there can be two distinct states q, q' of the system such that $\Gamma_?(q) = \Gamma_?(q')$. Meanwhile, based on this new notion of an attacker, we redefine the various notions introduced in the previous sections in the classical framework. An *observation* of the attacker is now a sequence from $\left((\Sigma_o \cup \{\epsilon\}) \times 2^{\Sigma_?}\right)^*$.

The set of observations of $\mathcal{A}_2$ that can appear when observing runs of G is denoted by $Tr_{\mathcal{A}_2}(\mathcal{G})$. Given an observable trajectory $\mu \in Tr_{\mathcal{A}_2}(\mathcal{G})$, the set of compatible sequences is given by: $[\![\epsilon]\!]_{\mathcal{A}_2} = \{\epsilon\}$ and

$$[\![\mu.(a,\gamma)]\!]_{\mathcal{A}_2} = \begin{cases} \{w \in [\![\mu]\!]_{\mathcal{A}_2}.a \cap L(G) \mid q_0 \xrightarrow{w} q \wedge \Gamma_?(q) = \gamma\}, \\ \qquad\qquad \text{if } a \in \Sigma_o; \\ \{w \in [\![\mu]\!]_{\mathcal{A}_2}.\sigma \cap L(G) \mid \sigma \in \Sigma_{uo}, q_0 \xrightarrow{w} q \wedge \\ \qquad\qquad \Gamma_?(q) = \gamma\} \text{ if } a = \epsilon. \end{cases}$$

Similar to the case of $\mathcal{A}_1$, the computation of the observer, starting from a subset of states X, can be defined inductively. Upon the arrival of a new observation (a, γ), the update of belief X by attacker $\mathcal{A}_2$ is defined by the function $\delta_{\mathcal{A}_2}(X, (a, \gamma))$:

- $\delta_{\mathcal{A}_2}(X, (\epsilon, \gamma)) = \Gamma_?(X', \gamma)$, where $X' = \{q' \mid \exists q \in X, a \in \Sigma_{uo}, (q, a, q') \in \delta\}$
- For every $a \in \Sigma_o, \delta_{\mathcal{A}_2}(X, (a, \gamma)) = \Gamma_?(X', \gamma)$, where $X' = \{q' \mid \exists q \in X, (q, a, q') \in \delta\}$

Then, for a sequence of observations, we can define:

- $\Delta_{\mathcal{A}_2}(X, \epsilon) = X$, and
- $\Delta_{\mathcal{A}_2}(X, \mu.(\sigma, \gamma)) = \delta_{\mathcal{A}_2}(\Delta_{\mathcal{A}_2}(X, \mu), (\sigma, \gamma))$ for every $\mu \in Tr_{\mathcal{A}_2}(G)$ and observation (σ, γ).

3.2 Attacker 3: input-aware attacker upon observation

We now model a type of attacker $\mathcal{A}_3$ who is aware of all allowed input events that are admissible only after an observable action $a \in \Sigma_o$ occurs. We still assume that inputs belong to $\Sigma_? \subseteq \Sigma_o$, and that the set of all allowed inputs is revealed after each observed action. Unlike attacker $\mathcal{A}_2$, attacker $\mathcal{A}_3$ does not update its belief when the system performs an unobservable move. Within this setting, $O_3 = 2^{\Sigma_?}$ and the function $obs_{\mathcal{A}_3}$ is given by:

$$obs_{\mathcal{A}_3}(a, q) = \begin{cases} (a, \Gamma_?(\delta(q, a))) & \text{if } a \in \Sigma_o \\ \epsilon & \text{otherwise.} \end{cases}$$

The set of observations of $\mathcal{A}_3$ that can appear during runs of G is defined as $Tr_{\mathcal{A}_3}(G) \subseteq (\Sigma_o \times 2^{\Sigma_?})^*$. Each observation in $Tr_{\mathcal{A}_3}(G)$ is a sequence of the form $(a_1, \gamma_1)(a_2, \gamma_2)\ldots$, where each a_i is an observable event and $\gamma_i \subseteq \Sigma_?$ corresponds to the set of input events that are admissible after the observation of the event a_i in the state of the system reached after executing a_i. Given an observation $\mu \in Tr_{\mathcal{A}_3}(G)$, the set of compatible sequences is given by:

$[\![\epsilon]\!]_{\mathcal{A}_3} = \{\epsilon\}$ and for $a \in \Sigma_o$

$[\![\mu.(a,\gamma)]\!]_{\mathcal{A}_3} = \{w \in [\![\mu]\!]_{\mathcal{A}_3}.a \cap \mathcal{L}(G) \mid q_0 \xrightarrow{w} q \wedge \Gamma_?(q) = \gamma\}$.

The refinement of a state estimate X by an observation (a, γ) is given by $\delta_{\mathcal{A}_3}(X, (a, \gamma)) = \Gamma_?(X', \gamma)$, where $X' = \{q' \in Q \mid \exists q \in X, \exists \rho = q \longrightarrow \ldots q', l(\rho) = a\}$. The state estimates of the attacker, starting from a set of states X, is recursively defined as follows:

- $\Delta_{\mathcal{A}_3}(X, \epsilon) = X$, and
- $\Delta_{\mathcal{A}_3}(X, \mu.(a, \gamma)) = \delta_{\mathcal{A}_3}(\Delta_{\mathcal{A}_3}(X, \mu), (a, \gamma))$, for $\mu \in Tr_{\mathcal{A}_3}(G)$ and $(a, \gamma) \in \Sigma_o \times 2^{\Sigma_?}$.

3.3 Attacker 4: an attacker informed of input changes

The last model of a passive attacker is informally defined as follows: we assume that the attacker $\mathcal{A}_4$ can continuously observe the possible inputs to the system, but is not aware of state changes. Hence, if two consecutive states q, q' in a run have the same inputs, and the move from q to q' produces no observable events, then $\mathcal{A}_4$ is not aware of any state change.

For $\mathcal{A}_4$, we have $O_4 = 2^{\Sigma_?}$ and $obs_{\mathcal{A}_4}$ is

$$obs_{\mathcal{A}_4}(a, q) = \begin{cases} (\epsilon, \Gamma_?(\delta(q, a))), & \text{if } a \in \Sigma_{uo} \wedge \Gamma_?(q) \neq \Gamma_?(\delta(q, a)) \\ \epsilon, & \text{if } a \in \Sigma_{uo} \wedge \Gamma_?(q) = \Gamma_?(\delta(q, a)) \\ (a, \Gamma_?(\delta(q, a)), & \text{if } a \in \Sigma_o. \end{cases}$$

The observations of $\mathcal{A}_4$ look like observations of $\mathcal{A}_2$, but with the difference that $\mathcal{A}_2$ obtains one pair (σ, γ) for each transition in a run. An observer $Det_{\mathcal{A}_4}(G)$, its transition relation $\delta_{\mathcal{A}_4}$ and $\Delta_{\mathcal{A}_4}$ can be built as for our previous attackers.

- $\delta_{\mathcal{A}_4}(X, (\epsilon, \gamma)) = \Gamma_?(X', \gamma)$, where $X' = \{q' \mid \exists \rho = q \xrightarrow{a_1} q_1 \ldots \xrightarrow{a_k} q_k \xrightarrow{a_{k+1}} q', l(\rho) = \epsilon \wedge \forall i \in 1..k, \Gamma_?(q) = \Gamma_?(q_i) \wedge \Gamma_?(q) \neq \Gamma_?(q')\}$
- For every $a \in \Sigma_o, \delta_{\mathcal{A}_4}(X, (a, \gamma)) = \Gamma_?(X', \gamma)$, where $X' = \{q' \mid \exists q \in X, (q, a, q') \in \delta\}$

Then, for a sequence of observations, we can define:

- $\Delta_{\mathcal{A}_4}(X, \epsilon) = X$, and
- $\Delta_{\mathcal{A}_4}(X, \mu.(\sigma, \gamma)) = \delta_{\mathcal{A}_4}(\Delta_{\mathcal{A}_4}(X, \mu), (\sigma, \gamma))$ for every $\mu \in Tr_{\mathcal{A}_4}(G)$ and an observation (σ, γ).

3.4 Attacker 5: an active attacker

We consider a slightly different attacker, namely one that is *active*. More precisely, attacker $\mathcal{A}_5$ learns whether an input is active only by trying to play it. This attacker

can observe actions from Σ_o, and also test whether an input is allowed at the current state by attempting to enter this input. If an input a is allowed at a state q, then the system moves to the next state $\delta(q,a)$. If the input is not allowed, then the system remains in state q and the attacker gets the information *not* a, indicating that the input was ineffective. We also consider that $\mathcal{A}_5$ can only test inputs to the system from the subset $\Sigma_{att} \subseteq \Sigma_? \cap \Sigma_o$.

To reflect the fact that an input is not fireable in a state we introduce a new alphabet $\Sigma_?^{not} = \{not\ \sigma | \sigma \in \Sigma_?\}$. In this setting, the alphabet of observed facts O_5 is given by $O_5 = \Sigma_o \cup \Sigma_?^{not}$, whereas the function $obs_{\mathcal{A}_5}$ is

$$obs_{\mathcal{A}_5}(a,q) = \begin{cases} \epsilon, & \text{if } a \in \Sigma_{uo} \\ (a,\epsilon), & \text{if } a \in \Sigma_o \setminus \Sigma_{att} \\ (a,a), & \text{if } a \in \Sigma_{att} \cap \Gamma_?(q) \\ (a, not\ a), & \text{if } a \in \Sigma_{att} \wedge a \notin \Gamma_?(q). \end{cases}$$

The set of observations is then defined as $Tr_{\mathcal{A}_5}(G) \subseteq (\Sigma_o \times (\Sigma_o \cup \Sigma_?^{not} \cup \epsilon))^*$, while the transition relation of $Det_{\mathcal{A}_5}(G)$ is:

$$\delta_{\mathcal{A}_5}(X,(a,\gamma)) = \begin{cases} \{q' \in Q \mid \exists \rho = q \longrightarrow \dots q', l(\rho) = a, \\ \qquad \text{if } a \in \Sigma_o \setminus \Sigma_{att} \wedge \gamma = \epsilon \\ \{q' \in Q \mid \exists q \in X, (q,a,q') \in \delta\}, \\ \qquad \text{if } a \in \Sigma_{att} \wedge \gamma = a \\ \{q \in X \mid \nexists q', (q,a,q') \in \delta\}, \\ \qquad \text{if } a \in \Sigma_{att} \wedge \gamma = not\ a. \end{cases}$$

As it was for the other attackers, the transition relation of observer $Det_{\mathcal{A}_5}(G)$ extends to words with $\Delta_{\mathcal{A}_5}(X, \mu.(a,\gamma)) = \delta_{\mathcal{A}_5}(\Delta_{\mathcal{A}_5}(X,\mu),(a,\gamma))$.

3.5 Comparing attackers

We can extend Corollary 1 to all attackers:

Proposition 2. For every $\mathcal{A}_i$, $i \leq 5$, S is opaque w.r.t. G and $\mathcal{A}_i$ iff $\forall \mu \in Tr_{\mathcal{A}_i}(G),\ \Delta_{\mathcal{A}_i}(X_0,\mu) \not\subseteq S$.

Let us now compare the discriminating power of attackers. We will say that attacker $\mathcal{A}_i$ is *less discriminating* than attacker $\mathcal{A}_j$, and write $\mathcal{A}_i \sqsubseteq \mathcal{A}_j$(with fixed alphabets $\Sigma, \Sigma_o, \Sigma_{att}, \Sigma_?$) iff for every automaton G and every set S of secret states, S is opaque w.r.t $\mathcal{A}_j$ implies that S is opaque w.r.t $\mathcal{A}_i$. Looking at the information collected by each type of attacker, we can state the following.

Proposition 3. We have the following relationships between attackers:

- $\mathcal{A}_1 \sqsubseteq \mathcal{A}_2$, $\mathcal{A}_1 \sqsubseteq \mathcal{A}_3$, $\mathcal{A}_1 \sqsubseteq \mathcal{A}_4$, and $\mathcal{A}_1 \sqsubseteq \mathcal{A}_5$
- $\mathcal{A}_3 \sqsubseteq \mathcal{A}_2$, $\mathcal{A}_4 \sqsubseteq \mathcal{A}_2$, and $\mathcal{A}_5 \sqsubseteq \mathcal{A}_2$

Obviously, $\mathcal{A}_1$ is the weakest attacker, as it only observes occurrences of actions in Σ_o, while all other observers have at least this information when an observable transition is executed. It might be surprising that $\mathcal{A}_5$, which is active, is less powerful than $\mathcal{A}_2$. Indeed, $\mathcal{A}_5$ can not always test *all* inputs of the system at the current state to get the same information as $\mathcal{A}_2$, since each test may force the system to move to another state.

4. ENFORCING OPACITY

If a system is not opaque, then one might try to control the system (or to restrict its behavior) so that the secret becomes opaque. This is achieved by dynamically forbidding or *disabling* certain events. Henceforth, we attach a *controller* to the system. This controller is in charge of deciding which inputs are allowed to occur or *enabled* at each point of an execution. We assume that the controller can observe all the events of Σ, but cannot control all of them. Thus, Σ, the alphabet of the system, is partitioned into Σ_c, the set of controllable events, and Σ_{uc}, the set of uncontrollable events. We assume that the controller has full knowledge of the system. Further, the controller is *belief-based*, i.e., it bases its decisions on its state estimate.

Definition 6. A *controller* is a function $\mathcal{C} : \mathcal{L}(G) \to 2^\Sigma$ that provides the set of actions that are enabled after a sequence $u \in \mathcal{L}(G)$, with the constraint that $\forall u \in \mathcal{L}(G), \mathcal{C}(u) \cap \Sigma_{uc} = \Gamma(\delta(q_o,u)) \cap \Sigma_{uc}$ (controllers cannot disable uncontrollable events). Given a state q, let $C_v(q)$ denote the set of valid control policies in q, i.e., all sets $c \subseteq \Gamma(q)$ that respect $c \cap \Sigma_{uc} = \Gamma(q) \cap \Sigma_{uc}$.

Intuitively, when a sequence u ends in a state q, then a controller for G can only use a policy in $C_v(q)$. A controller $\mathcal{C}$ can be encoded by an automaton $\mathcal{A}_\mathcal{C} = (Q_C, q_{0_C}, \delta_C, \Sigma, \lambda_C)$, where Q_C is a not-necessarily finite set of states, δ_C is a deterministic transition relation, and $\lambda_C : Q_C \to 2^\Sigma$ associates each state with a set of enabled events.

Definition 7. Let G be an automaton, and let $\mathcal{C} : \mathcal{L}(G) \to 2^\Sigma$ be a controller, described by its automaton $\mathcal{A}_\mathcal{C}$. Then $G/\mathcal{C} = (Q_{G/\mathcal{C}}, (q_0, q_{0_C}), \Sigma, \delta_{G/\mathcal{C}})$ is the automaton obtained by taking the synchronous product of G with $\mathcal{A}_\mathcal{C}$, inductively built as follows.

- The initial state is (q_0, q_{0_C})
- If $(q,q_C) \in Q_{G/\mathcal{C}}$, $\delta(q,a) = q'$, $a \in \lambda_C(q_C)$ and $\delta(q_C,a) = q'_C$, then $((q,q_C),a,(q',q'_C)) \in \delta_{G/\mathcal{C}}$, and (q',q'_C) belongs to $Q_{G/\mathcal{C}}$.

One can notice that $\mathcal{L}(G/\mathcal{C}) \subseteq \mathcal{L}(G)$. A question that immediately arises: how is the knowledge of an attacker affected by control? We assume that attackers do not know the control applied to the system. Nevertheless, this does not mean that control does not affect observations. For instance, consider a situation where the current state is q, $\Gamma(q) = \{?a, ?b\}$, but the controller disables $?a$ at q. We consider that the attackers get information about the actual set of available actions, i.e., when some control is in use. An attacker will receive information of the form $(e, \{?b\})$ when the system enters state q, where e is either ϵ or an action name. Let X be the current state estimate of the attacker. It can be refined by the observation, but recall that some actions that are not feasible from current state might actually be forbidden by a controller. Hence we define $Refact(X,\gamma) = X \setminus \{q \in Q \mid \exists e \in \gamma, \nexists q', q \xrightarrow{e} q'\}$. This refinement of state estimate removes from X all states from which actions from γ are not fireable. We can now define the control and synthesis problem:

Control Problem: *Given a system G, a secret S, an attacker $\mathcal{A}$ that observes Σ_o, and a controllable alphabet $\Sigma_c \subseteq \Sigma$, is there a controller $\mathcal{C}$ such that S is opaque for $\mathcal{A}$ in $G/\mathcal{C}$?*

Synthesis Problem: *Given a system G, an attacker $\mathcal{A}$, and a secret S such that a controller with controllable alphabet $\Sigma_c \subseteq \Sigma$ exists to keep S opaque for $\mathcal{A}$, synthesize a map $\mathcal{C} : \mathcal{L}(G) \to 2^\Sigma$ such that S is opaque for $\mathcal{A}$ in $G/\mathcal{C}$.*

In the rest of this section we will show that for passive attackers $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ and $\mathcal{A}_4$, controllers are indeed

strategies in finite co-reachability games. Within these games, positional strategies are sufficient to win, yielding finiteness of controllers.

Definition 8. A two-player arena with partial information is a tuple $H = (V, v_0, \mathcal{M}, \longrightarrow, \mathcal{O}_0, \mathcal{O}_1)$, where V is a set of positions, partitioned into $V_0 \uplus V_1$ (depicting player 0 and player 1's nodes, respectively); v_0 is the initial node of the arena; $\longrightarrow \subseteq V \times \mathcal{M} \times V$ depicts moves in the arena; and $\mathcal{O}_i : V \to \Psi_i, i \in \{0, 1\}$ are observation functions, mapping vertices to a finite arbitrary observation alphabet. This can be used to model partial knowledge on the current vertex of the arena. Player i has to choose a move from a node in V_i. A two-player co-reachability game is given by an arena and a winning condition $Win \subseteq V$. Player 0 wins the game if it can prevent the game from reaching a position in Win.

A *play* in an arena is either a finite sequence $\pi = v_0.v_1.v_2 \dots v_k$ or an infinite one $\pi = v_0.v_1.v_2 \dots v_k \dots$. We denote by $\pi(j)$ the j^{th} vertex of play π. A play π is *winning* for player 1 (and losing for player 0) with the co-reachability condition Win if there exists j such that $\pi(j) \in Win$. A *strategy* for player i is a map $\sigma_i : V^*.V_i \to 2^{Act_i}$, where Act_i is the set of moves of player i. In particular, we require that

- for every play $\pi = v_0.v_1 \dots v_k$ with $v_k \in V_i$, $\sigma_i(\pi) \subseteq Act(v_k)$, where $Act(v_k)$ is the set of legal moves from node v_k in the arena.
- for every pair of plays π, π' such that $\mathcal{O}_i(\pi) = \mathcal{O}_i(\pi')$, $\sigma_i(\pi) = \sigma_i(\pi')$ (i.e., strategies are consistent with observations).

A play is *consistent* with strategy σ_i iff for every $\pi(j) \in V_i$, $\pi(j+1) \in \sigma_i(\pi(0) \dots \pi(j))$. Player i *wins* in position v iff it has a strategy σ_i such that for every strategy σ_{1-i} of player $1-i$, all plays starting from v and consistent with strategies σ_i, σ_{1-i} are winning for player i.

We will say that a two-player game is a game of perfect information iff $\forall v \in V, O_1(v) = O_2(v) = \{v\}$. Otherwise, the games are called games with partial information. It is known that perfect information games with (co)reachability, Büchi, or parity objectives are determined : from each vertex, one (and only one) player has a winning strategy. Further, perfect information games need only positional strategies (Grädel and Thomas, 1998). It is known that players have a winning strategy in a partial information game iff they have a belief-based strategy (Chatterjee and Doyen, 2010). Consequently, to decide whether a game is winning for player i, it is sufficient to build an arena containing the representation of the knowledge that each player has of the possible current state of the system. Then, solving a full-information game on this arena allows for the computation of belief-based strategies.

We are now ready to define a game that allows us to solve the opacity control and synthesis problem for each type of attacker. As attackers can only rely on their beliefs to reach a state estimate that is contained in S, and do not know the current state of the system, this will be a partial information game. We begin with attackers of type $\mathcal{A}_2$.

4.1 A controller for attackers of type $\mathcal{A}_2$

In this game, the controller is player 0, the rest of the system is the player 1, and the attacker of the system is used to determine a winning condition on states. It might seem strange *a priori* to consider the system as an opponent to the controller, and not to the attacker. The reason for this is that attackers are passive. Hence they cannot guide the system to play the best actions leading to information leakage. However, to ensure that a secret is preserved, one can assume that the system, even if not malevolent, will always play the worst choice with respect to secrecy.

Let G be a system over alphabet Σ, $S \subseteq Q$ be a set of secret states, q_0 be an initial state, Σ_o be the set of events observed by attacker $\mathcal{A}_2$, and Σ_c be a set of controllable events. The arena for this opacity control and synthesis problem is defined as a tuple $H_{G,S,\mathcal{A}_2} = (\mathcal{V}, v_0, \mathcal{M}, \longrightarrow_H, \mathcal{O}_0, \mathcal{O}_1)$ where:

- $\mathcal{V} = \mathcal{V}_0 \cup \mathcal{V}_1$ is a set of nodes. We have $\mathcal{V}_1 = Q \times 2^Q \times 2^\Sigma$. This set of nodes denotes positions of player 1 (the system). $\mathcal{V}_1$ contains nodes of the form (q, X, Σ'), where q is the current state of G, X is the attacker's estimate of the set of possible states according to its observations, and $\Sigma' \subseteq \Sigma$ is the last choice of the controller, i.e., a set of actions that the controller allows from state q. Similarly, $\mathcal{V}_0 = Q \times 2^Q \times \Sigma$ is the set of positions of player 0 (i.e., the controller). Every node $(q, X, a) \in \mathcal{V}_0$ represents the next current state q and the next possible state estimate X after performing action a. The initial node of the game is node $v_0 = (q_0, \{q_0\}, \epsilon) \in \mathcal{V}_0$, i.e., the system starts from its initial node, the attacker has perfect knowledge of it, and it is then up to the controller to decide which actions are allowed from the initial state. In this node, ϵ symbolizes the fact that no action has yet been performed by the system.
- $\mathcal{M} = \mathcal{M}_0 \cup \mathcal{M}_1$ is the alphabet of the game, where $\mathcal{M}_0 = \Sigma \times 2^\Sigma$ represents moves of the controller, which makes a choice between the admissible control policies, and $\mathcal{M}_1 = \Sigma$ represents moves of the system.
- $\mathcal{O}_0(v) = \mathcal{O}_1(v) = v$, i.e., both players have perfect knowledge of the game's positions.
- Moves of the arena are a subset $\rightarrow_H \subseteq (\mathcal{V}_1 \times \mathcal{M}_1 \times \mathcal{V}_0) \cup (\mathcal{V}_0 \times \mathcal{M}_0 \times \mathcal{V}_1)$ and are built as follows:
  i) For every $c_i \in C_v(q_o)$, $(v_0, c_i, (q_0, \{q_0\}, c_i))$ is a move of $\rightarrow_H$. Note that $(q_0, \{q_0\}), c_i) \in \mathcal{V}_1$.
  ii) If $((q, X), c)$ is a node of $\mathcal{V}_1$, and $a \in c$ is an action allowed by the controller (and, hence, allowed from q), then $((q, X, c), (\delta(q, a), X, a)) \in \rightarrow_H$.
  iii) Given a node $(q, X, a) \in \mathcal{V}_0$, transition $((q, X, a), (a, c), (q, X', c)) \in \rightarrow_H$ for all $c \in C_v(q)$. Let $\gamma = c \cap \Gamma_?(\delta(q, a))$, and $X_{O,a} = \Delta_o(X, a)$. Then we have $X' = Refact(X_{O,a}, \gamma)$, i.e., the attacker's knowledge is updated, and includes states that are reachable from X when observing a, and which allow inputs from γ. Note that as the attacker is not aware of c_i, X' is the only deduction that an attacker of type $\mathcal{A}_2$ can do. (See Fig. 1)

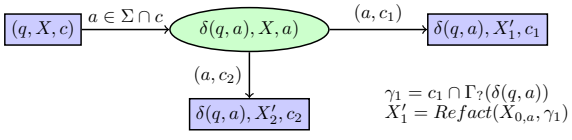

Fig. 1. A part of the arena for the opacity game

Fig. 1 shows several possible types of moves in arena H. Nodes of the controller (player 0) are represented as circles, and nodes of player 1 as rectangles. In node (q, X, c), the system decides to fire action $a \in \Sigma \cap c$. The system then moves to a position $(\delta(q, a), X, a)$ meaning that the system will move to position $\delta(q, a)$. From this node, the controller has to decide which actions will be fireable from the next state $\delta(q, a)$, which will impact what is observable by the attacker. In Fig. 1, the controller has two possible choices, namely c_1 and c_2. After a choice of the controller the move is performed and the attacker is informed of the move. Following the definition of an attacker of type $\mathcal{A}_2$, the observation produced is of the form (e, γ), where $e = \epsilon$ if the action played by the system is unobservable, and $e = a$ if the performed action is observable. Similarly, when control c_i is chosen, the set of possible inputs that attacker $\mathcal{A}_2$ can observe is $\gamma = \Gamma_?(\delta(q, a)) \cap c_i$. The state estimate of the attacker is updated accordingly, i.e., upon the choice of control c_i the state estimate becomes $X' = Refact(\Delta_o(X, a), \gamma)$.

The objective of the controller is to avoid states of the form (q, X, c), where $X \subseteq S$, i.e., player 1 wins if it can force the system to one of these states. This is a classic turn-based co-reachability game, where the controller needs to choose valid control policies that avoid positions where the attacker's knowledge is contained in S. It is well-known that perfect information (co)-reachability games are positional and determined, and can be solved in linear time by computing an attractor for the set of bad states that player 0 wants to avoid. It remains to show that a controller for opacity exists iff there is a wining strategy in this reachability game.

Theorem 1. Let G be an automaton, S be a secret, and $\mathcal{A}_2$ be an attacker. Secret S can be made opaque by some controller $\mathcal{C}$ iff player 0 has a winning strategy in the co-reachability game over arena $H_{G,S,\mathcal{A}_2}$ with winning condition $Win = Q \times 2^S \times 2^\Sigma$.

Note that belief-based controllers are not the only possible controllers. Consider the automaton of Fig. 2, where $\Sigma = \{a, u, c\}$ and $\Sigma_o = \{a, c\}$. To avoid reaching secret state S, a controller can forbid action a when the difference between the number of as and cs is equal to k. Such a controller never computes the state estimate of an attacker, but yet is sufficient to guarantee opacity.

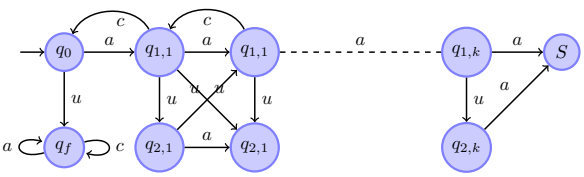

Fig. 2. An automaton G that admits simple controllers

Corollary 3. For an alphabet of fixed size, enforcement of opacity for an attacker of type $\mathcal{A}_2$ is in EXPTIME.

As already mentioned, positional strategies for two-player games with perfect information are sufficient. It is hence sufficient to design strategies that associate a decision to each node of the arena. Recall, however, that our games are played on an arena that represents the attacker's knowledge. This gives us the following corollary.

Corollary 4. Finite state (belief-based) controllers are sufficient to ensure opacity. Opacity controllers are of size in $O(|G|.2^{|G|})$. There exist automata of size in $O(n)$ in which opacity is ensured only by controllers of size in $O(n.2^n)$.

4.2 A controller for attackers of type $\mathcal{A}_3$ and type $\mathcal{A}_4$

The technique used in Theorem 1 for attackers of type $\mathcal{A}_2$ can be easily adapted for attackers of type $\mathcal{A}_3$ and $\mathcal{A}_4$, and the opacity enforcement game remains a perfect information reachability game. The construction of the arena is very similar: the major difference resides in how the attacker's state estimate is updated. With attacker $\mathcal{A}_3$, for instance, a move from node $((q, X), c) \in \mathcal{V}_1$ with action a leads to a node $((\delta(q, a), X, a)$, as was the case for attacker $\mathcal{A}_2$. Moves from nodes of the controller (nodes of $\mathcal{V}_0$) of the form (q, X, a) lead to nodes of the form (q, X, c) if a is unobservable, and otherwise to nodes of the form (q, X', c), where $X' = Refact(X_o, c \cap \Gamma_?(\delta(q, a))$ is an update of the knowledge of $\mathcal{A}_3$, with $X_o = \Delta_o(X, a)$.

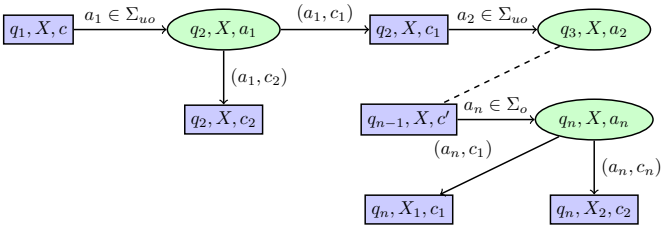

Fig. 3. Part of the arena for attacker $\mathcal{A}_3$

For attacker $\mathcal{A}_4$, the knowledge update is performed as soon at the input set changes. From a node of the form $(q, X, c) \in \mathcal{V}_1$ the system can choose an action a allowed by the control c, and move to a state $(\delta(q, a), X, a, q)$. Notice that the state that is left is remembered. From a node $(q', X, a, q) \in \mathcal{V}_0$ when choosing a control c, one moves to node (q', X, c) if $\Gamma(q) = \Gamma(q')$ and a is unobservable, and to node (q', X', c) otherwise, where X' is computed as for attackers of type $\mathcal{A}_2$ and $\mathcal{A}_3$. Opacity enforcement games remain unchanged on this slightly-modified arena.

5. ENFORCEMENT WITH ACTIVE ATTACKERS

With an active attacker $\mathcal{A}_5$, the game setting changes. Since attackers are active, they can have a strategy to drive the system into configurations leaking the secret S. In the previous games designed to solve opacity for passive attackers, the partial observation of such an attacker is only used to designate which nodes of the arena are losing states for the controller. Observation functions for this kind of game are perfect information functions. Active attacker $\mathcal{A}_5$, on the other hand, can take decisions, and is now one of the players in the arena, although it has only a partial information on the actual state of G. We assume that the attacker is one of the processes of the system, and owns a subset of actions $\Sigma_{att} \subseteq \Sigma_o$ that are performed only by him. These inputs can be performed/tested by the attacker immediately after each observable action. Yet, in this setting, the rest of the system can still perform actions that help $\mathcal{A}_5$ acquire further information. We will hence distinguish nodes where the system plays, and nodes where the attacker chooses to input something to the system. This setting forces us to consider a new type of game, namely, three-player games, as introduced in (Chatterjee and Doyen, 2014).

Definition 9. A *three-player game* is a tuple $\mathcal{H} = (Q, q_0, \delta, O_{i,i\in 1..3}, obs_{i,i\in 1..3})$, where Q is a set of nodes, q_0 an initial node, and $\delta : Q \times A_1 \times A_2 \times A_3 \to Q$ is a deterministic function that gives the successor state $\delta(q, a_1, a_2, a_3)$ of q when players $1, 2, 3$ concurrently play actions a_1, a_2, a_3. The game is *turn-based* if for every state q and every triple in $A_1 \times A_2 \times A_3$ the next state depends only on the action chosen by one of the players. For every player, $O_i \subseteq 2^Q$ is a partition of the set of states, and $obs_i : Q \to O_i$ is the function that assigns to q the unique observation of player i that contains q. A winning reachability condition is a subset of states $Win \subseteq Q$. An observation-based strategy for player i is a map $\sigma_i : Q \times O_i^* \to Q$. Given a game arena $\mathcal{H}$ and a winning condition Win, the three-player reachability problem is to decide if $\exists\sigma_1, \forall\sigma_2, \exists\sigma_3$ such that the play that satisfies strategies $\sigma_1, \sigma_2, \sigma_3$ visits Win at least once.

We define an arena $\mathcal{H}_G^{active} = (V, v_0, \delta, O_i, obs_i)$ as follows. The set of nodes V is partitioned into $V = V_1 \cup V_2 \cup V_3$, where V_1 represents positions of the attacker $\mathcal{A}_5$, V_2 are positions of the controller, and V_3 are positions of the system.

$$V_1 = \{(q, X, c, att) \mid X \in 2^Q, C \in 2^\Sigma\}$$
$$V_2 = \{(q, X, a, q') \mid q, q' \in Q, X \in 2^Q, a \in \Sigma\}$$
$$V_3 = \{(q, X, c, sys) \mid X \in 2^Q, C \in 2^\Sigma\}$$

We set $v_0 = (q_0, \{q_0\}, \epsilon, q_0) \in V_2$, i.e., the controller starts (as no action is played yet, the third component of the node is simply ϵ). The transition relation δ is then computed as follows: the game allows only moves from V_1 to V_2, from V_3 to V_2 and V_1 and from V_2 to V_3. It is turn based, i.e., for every node $n \in V_i$, $\delta(n, a_1, a_2, a_3)$ only depends on a_i.

There exists a move from $(q, X, c, sys) \in V_3$ to $(q, X, c, att) \in V_1$ if $\Gamma_?(q) \cap c \neq \emptyset$ (if we consider that $\Gamma_?(q) \subseteq \Sigma_o$) [2]. This move models the situation where the system does not perform an action, some inputs are feasible, and the system lets the attacker test them. There exists a move from $(q, X, c, sys) \in V_3$ to $(q, X, a, \delta(q, a)) \in V_2$ if $a \in c$. This move models the system's decision to perform an action. Right after this decision, upon reaching the destination state, the controller has to take a control decision for inputs in Σ_c. As with games that have attackers of type $\mathcal{A}_2$ and $\mathcal{A}_3$, knowledge of $\mathcal{A}_5$ is not updated by this move, which allows the controller to choose the next control decision.

A move from $(q, X, c, att) \in V_1$ to $(q, X, a, \delta(q, a)) \in V_2$ exists if $a \in c \cap \Gamma_?(q)$. This move models input to the system performed by the attacker, and the consequences of this input (i.e., a state change). There exists a move from $(q, X, c, att) \in V_1$ to $(q, X', c, sys) \in V_2$ if $a \notin c \cap \Gamma_?(q)$, with $X' = X \setminus \{q \in X \mid a \in \Gamma_?(q)\}$ if $a \in \Sigma_{uc}$, and $X' = X$, otherwise. Let us detail the computation of X'. Node (q, X, c, att) is an attacker node, i.e., from this node, an attacker may want to test an input to the system to refine its estimate of the system's state. If this input $?a$ fails, then it is either because this input is not feasible from the current state, or because it was disabled by the controller. If the input is controllable, an attacker cannot distinguish between a state q_f such that $?a \notin \Gamma_?(q_f)$ and a state q_a such that $?a \in \Gamma(q_a)$ but $?a \notin c$. If the attacker estimates that the system can be in state X before testing $?a$, and subsequently learns that this input is not feasible, then its knowledge remains unchanged. On the other hand, if the input does not succeed, and $?a$ is not controllable, then X can be refined, since the current state of the system is necessarily a state q with $?a \notin \Gamma(q)$.

The controller allows initial moves from node v_0 to every node of the form (q_0, c, sys) where $\Gamma(q_0) \cap \Sigma_{uc} \subseteq c \subseteq \Gamma(q_0)$. All moves from a control node of the form $n = (q, X, a, q') \in V_2$, with $a \notin \Sigma_o$, are moves to nodes of the system of the form $(q', X, c, sys) \in V_3$, with $c \in 2^\Sigma$. Notice that the knowledge of the attacker is not updated. This models the fact that unobservable moves do not bring information to the attacker. All moves from a control node of the form $n = (q, X, a, q')$, with $a \in \Sigma_o$ are moves of the form $(q', X', c, sys) \in S_3$, with $c \in 2^\Sigma$, and $X' = \Delta_o(X, a)$. The knowledge of the attacker is updated only according to the observation of action a, whereas for attacker $\mathcal{A}_5$, the observation of allowed inputs is performed actively, i.e., via a test made by the attacker.

We can now define the observation functions. The controller (player 2) has perfect knowledge of the system's state, so we define $O_2(n) = n$ for every node of the system. The system (player 3) also has perfect information on the current node of the system, so $O_3(n) = n$. The attacker only has partial information, i.e., if $n = (q, X, x, sys)$, then $O_2(n) = X$. Thus, the attacker cannot differentiate states containing the same state estimate. Lastly, the winning condition is $Win_S = \{n \mid O_2(n) \subseteq S\}$.

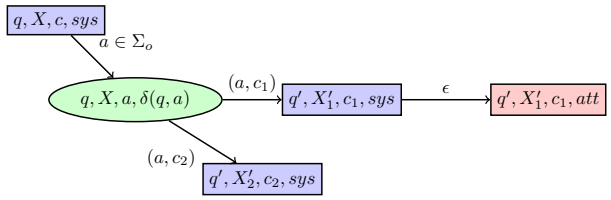

Fig. 4. An arena for the opacity game with an active attacker $\mathcal{A}_5$. States of the attacker are represented as red rectangles, states of the controller are green circles, and states of the system are blue rectangles.

Theorem 2. Let G be an automaton and S be a secret. The opacity of S w.r.t. an active attacker $\mathcal{A}_5$ can be enforced by a belief-based controller iff player 2 has a winning strategy σ_2 against all strategies σ_1, σ_3 of players 1 and 3 in the three-player reachability game over arena H_G^{active} with winning condition Win_S.

In their most general form, three-player games of partial information are undecidable (Reif, 1979), but Chatterjee and Doyen (2014) showed that three-player games are decidable when player 1 is less informed than player 2. This is the case in our setting, as the attacker only has partial information on the current location in the arena, while players 2 and 3 have perfect knowledge of it. More precisely, Chatterjee and Doyen (2014) show that three-player games are 2-EXPTIME complete, in general. The main idea of their proof is to show a reduction from a three-player game to an exponentially larger partial information two-player game with the same objective. Since partial information two-player games can be solved in EXPTIME, this yields the 2-EXPTIME upper bound for complexity of three-player games in general. A remark

[2] Otherwise, we should write $\Gamma_?(q) \cap c \cap \Sigma_o \neq \emptyset$

in Chatterjee and Doyen (2014) is particularly interesting: if player 2 has perfect information on the game, then the decision procedure is only in EXPTIME. We can even avoid an additional exponential blowup, as explained in the following corollary.

Corollary 5. Given an automaton G, a set of secret states S and an active attacker $\mathcal{A}_5$, deciding existence of a controller ensuring opacity of G wrt active attacker $\mathcal{A}_5$ can be solved in EXPTIME.

6. CONCLUSION

We have proposed a unified setting for opacity with passive and active attackers. For all attackers, opacity enforcement can be brought back to a partial-information game.

Related work : Opacity can be seen as the opposite of diagnosis problems. In diagnosis (Sampath et al., 1995), the question of interest is whether fault occurrences can be detected with partial observation of a running system. This question boils down to checking that there exists no pair of observationally-equivalent runs, one of which is faulty, the other one that is not. Ensuring diagnosability is a desirable property of a system. The goal of opacity, on the other hand, is to prevent an attacker from gaining certainty about properties of executed runs by making such runs observationally equivalent to a run that does not possess the property of interest. From an algorithmic point of view, techniques to ensure diagnosability consist of verifying the absence of pairs of runs, which can be done in polynomial time on the self-product of the specification. *A contrario*, to verify opacity we need to check properties of a whole equivalence class for traces, and not only a pair of runs. This requires modeling the complete knowledge of an attacker, and immediately yields an exponential blowup.

Opacity has been addressed in several settings since the work of Bryans et al. (2005) and Badouel et al. (2007). Our approach does not only consist of checking whether opacity holds, but also provides the means to control a system in such a way that observed traces are not sufficient to decide whether the current state of the system is a secret one. Another enforcement technique was proposed by Cassez et al. (2012). This approach consists of dynamically changing the observation of an attacker using *masks* that hide a subset of transition labels that would otherwise be seen by the attacker. In this setting, the behavior of the system remains unchanged, but the control dynamically changes the way the system is observed. Our approach differs, as we forbid some behaviors. In fact, the approaches are orthogonal: one can easily design a system that is opaque with the help of masks, and not with control (e.g., when all events are uncontrollable). Conversely, one can easily design a system where masks are useless but where control can make a system opaque (for instance, when masks cannot reduce the set of observable events). Another technique to guarantee opacity is to allow systems to produce fictitious events as in Wu and S. Lafortune (2014). This technique is orthogonal to our setting but it should possible to model games where an "active" systems that can insert dummy observations.

Future work: This work can be extended in several ways. First of all, it should be straightforward to consider enforcement of opacity for regular properties. Indeed, as soon as runs satisfying a property P are recognized by an automaton $\mathcal{A}_P$, enforcing opacity of runs of G satisfying P simply means enforcing state-based opacity in $G \times \mathcal{A}_P$. Other open questions deal with the power of attackers. So far, we have considered that the attacker is alone, and can observe and interact with the system. An interesting question is whether this setting can be extended to a coalition of players that run distributed attacks. We assumed that controllers of the system had full knowledge of the current state of the system. This allowed us to recast opacity control in a game setting with perfect observation for the controller. Now, if the controller is a particular component of the system, and observes only a subset of actions performed and has only partial information on the current global state of the system, the setting for enforcement questions changes. For passive attackers, the enforcement games become two-player games *with partial information*, which are exponentially harder to solve than perfect information games. In an active setting, opacity enforcement becomes a three-player game with partial information. Such games are usually undecidable Peterson and Reif (1979), but there is still a chance that opacity enforcement problems can be recast in a decidable setting of (Chatterjee and Doyen, 2014).

REFERENCES

Badouel, E., Bednarczyk, M.A., Borzyszkowski, A.M., Caillaud, B., and Darondeau, P. (2007). Concurrent secrets. *Discrete Event Dynamic Systems*, 17(4), 425–446.

Bérard, B., Mullins, J., and Sassolas, M. (2015). Quantifying opacity. *Mathematical Structures in Computer Science*, 25(2), 361–403.

Bryans, J., Koutny, M., Mazaré, L., and Ryan, P.Y.A. (2005). Opacity generalised to transition systems. In *Formal Aspects in Security and Trust*, 81–95.

Cassez, F., Dubreil, J., and Marchand, H. (2012). Synthesis of opaque systems with static and dynamic masks. *Formal Methods in System Design*, 40(1), 88–115.

Chatterjee, K. and Doyen, L. (2010). The complexity of partial-observation parity games. In *Proc. of LPAR'17*, volume 6397 of *LNCS*, 1–14.

Chatterjee, K. and Doyen, L. (2014). Games with a weak adversary. In *Proc. of ICALP 2014, Part II*, volume 8573 of *LNCS*, 110–121.

Goguen, J. and Meseguer, J. (1982). Security policies and security models. In IEEE (ed.), *Proc of IEEE Symposium on Security and Privacy*, 11–20.

Grädel, E. and Thomas, W. (1998). *Automata Logics, and Infinite Games : A Guide to Current Research*, volume 2500 of *LNCS*.

Peterson, G.L. and Reif, J.H. (1979). Multiple-person alternation. In *FOCS'79*, 348–363. IEEE.

Reif, J. (1979). Universal games of incomplete information. In *STOC*, ACM Press.

Sampath, M., Sengupta, R., Lafortune, S., Sinaamohideen, K., and Teneketzis, D. (1995). Diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 40(9), 1555–1575.

Wu, Y. and S. Lafortune, S. (2014). Synthesis of insertion functions for enforcement of opacity security properties. *Automatica*, 50(5), 1336–1348.